DIUS BOWLING CLUB

### PREPARED FOR

DIUS

### PREPARED BY

SHAILENDAR.S

OVERVIEW

The task is to come up with a solution to implement a scoring system for the game of Bowling programmatically. The scoring system needs to calculate the score based on the number of pins rolled per strike. There are 10 rounds per game and there are 2 strikes per round. The scoring system is designed to calculate the score for a single player at a time with additional bonus points for certain scenarios.

The approach to implement the solution for the task will be to break down the problem into simple tasks. Identify the rules and validations that need to be considered while solutioning. Visualize the business logic and the business flow from a technical implementation point of view. Identify the parts of the problem which needs to be stored and maintained. Identify my technical limitations that I need to keep in mind while coming up with the solution.

The approach is as below.

APPROACH

The first task in hand is to break down the requirement into simple tasks. The requirement can be broken down into below tasks.

1. The Game flows.
   1. Note down the Round no and start with 1.
   2. Note down the strike no and start with 1.
   3. Note down the score and start with 0
   4. Record the point scored in the first strike
   5. Check if the point scored in the first strike is 10
   6. If the point scored is 10, then move to the next round and ignore the second strike.
   7. Total score will be score plus the number of pins rolled in a round
   8. If it is less than 10 then note down the points score in the second round
   9. Add the round no to plus 1 and continue till 10
   10. If the score in the first or second strike is 10 in the 10th round then add one more round as a bonus
   11. If the sum of the score in strike 1 and strike 2 is 10 in the 10th round then add one more strike as a bonus.
   12. Calculate the total score and end the game.
2. The Points Calculation.

The points calculation is a simple sum of scores but this math will change under certain scenarios. Those scenarios are listed below along with the approach of the points calculation for these scenarios. Consider S1 as the points scored in strike 1 and S2 as the points scored in strike 2 The calculation of points will be as below.

* 1. If the round was a normal round and the number of pins rolled in each strike is less than 10

Score = score + S1+S2

* 1. If the previous round was a strike (all 10 pins bowled in a single strike in the previous round)

Score = Score + (S1+S2) \*2

* 1. If the previous round had a spare (all 10 pins bowled in two strikes in the previous round)

Score = Score + (S1\*2 + S2)

* 1. If there is a strike for 2 consecutive rounds

Score = Score + S1+(S1+S2) \*2

* 1. If there is a strike in the final round and the strike in the ninth round

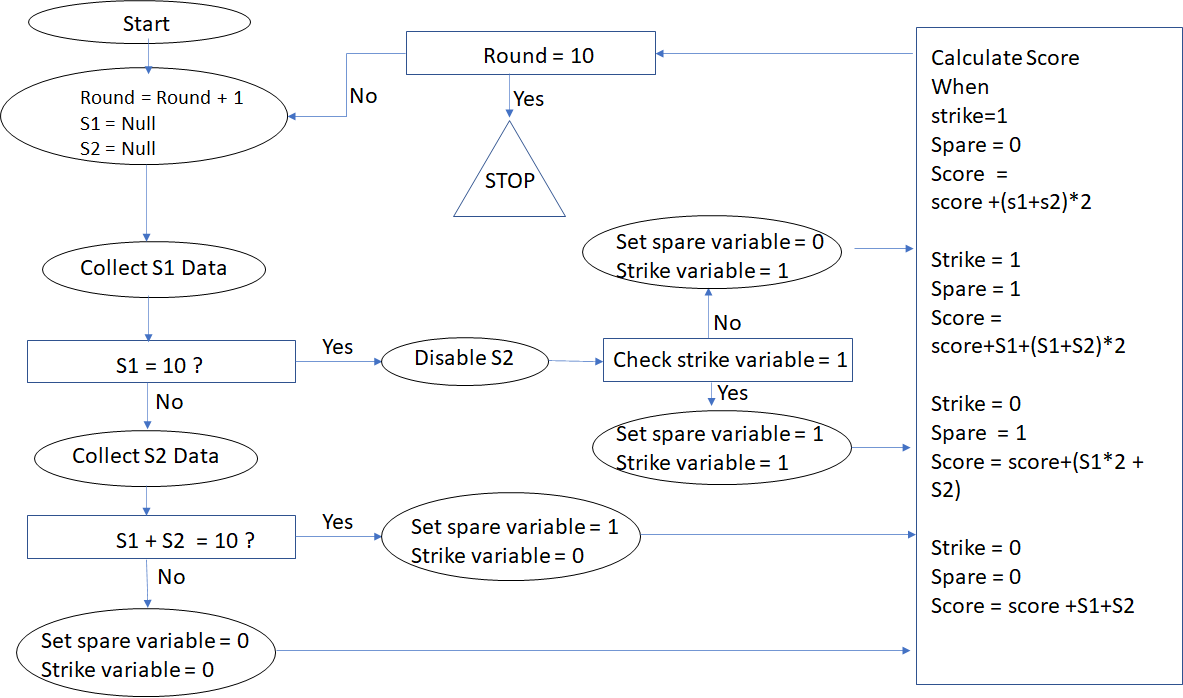
Score = Score + (S1\*2 +S2)

* 1. If there is a spare in the final round and the strike in the ninth round

Score = Score + S1

1. The validations in the game.
   1. The options for the number of pins for the second strike should always be 10 minus the number of pins rolled in the first strike
   2. The option for second strike should not be allowed for the scenario when there is a first strike.
   3. The maximum score cannot be more than 300
   4. The maximum number of rounds should be 10 and 11 in case of a strike or spare in the final round

Below is a simple algorithm of the tasks excluding the final round logic



In any given technology the decision makers will be the conditions that need to be in place. The decisions are highlighted in the rectangle shape barring the score calculation. The score calculation will be a method which will receive the data of the pins rolled and the previous strike or spare value. The other functionality will be to collect the data from the user and apply the business logic on top of it. In general, a **getPinsrolled** method will collect the data from the user assuming the number of pins for strike 2 is handled in the front end. The **getpinrolled** will check if the current roll is a strike or a spare and set the variable accordingly and then pass on to the **calculatescore** method. The calculate score method will calculate the score based on the scenario it fits in and resets the pin values to 0 to get the fresh data from the next strike.

IMPLEMENTATION

I have implemented the above logic in react.js technology. Since we are not worried about storing the final score or the entire strike history for all the 10 rounds, we don’t require a backend setup.

All the variables involved above are maintained as state variables. The project is created under three layers.

**The Top layer or a container:**

This ideally holds the variables required for the application as states. This contains the calculation logic and the controls to reset the values to null for the next round. This layer also gets the html components from the last layer through the middle layer and renders it to the browser.

**The Last layer or a UI:**

The last layer has all the html components defined for the application. For the bowling points calculation application, I used a single dropdown to select the number of pins rolled. This single dropdown is rendered twice to select the points for the 2 strikes that are made for a round. This layer captures the value provided by the user and passes on to the top layer for calculation.

**The middle layer or a component:**

The middle layer is responsible for redirecting the data to the appropriate function defined in the top layer. The middle receives the input from both the top and the bottom layer and applies the necessary business logic on top of it and passes on to the appropriate function to be processed further. An example to explain the role of the middle layer is as below.

There is a single drop down used to select the pins rolled for both the strikes. If the pins rolled in the first strike is 5 the middle layer ensures that the options provided to select the number of rolls rolled in the second strike should be a maximum of 5.

The link to the application is below:

<https://github.com/ShailyDeveloper/Dius>

CHALLENGES

There were few challenges while implementing the 10th round logic. I had to implement the logic without disturbing the existing ones. The last round logic was all together a new requirement and need to be called only under the scenario of last round spare or a strike. The calculation part for this was written separately without disturbing the existing score calculation. Things got little tricky for a scenario when the 9th round is a spare or strike followed by a 10th round spare or strike. However, this has been implemented by redirecting the last round event into a separate logic within the calculate score method.